US-PAT-NO: 6442620

DOCUMENT-IDENTIFIER: US 6442620 B1

TITLE: Environment extensibility and automatic services for component applications using contexts, policies and activators

--------- KWIC ---------

Application Filing Date - AD (1):
19980817

Brief Summary Text - BSTX (7):
The object execution environments of the above mentioned COM, OLE, DCOM, and MTS systems have several behaviors towards component application objects that depend on locality or like environment aspects (hereafter referred to as "domains") of the component application objects. The domains of a component application object in these environments include physical location (e.g., machine and process), isolation domains (e.g., process, user/kernel, security, transactions, auditing), synchronization domains (e.g., apartments, **activities**), object lifetime and identity domains (e.g., persistence, just-in-time activation, assemblies, per-client global and shared state), and representation domains (e.g., unicode/ANSI, 16/32-bit, locale, native/automation, Java/COM). For example, the OLE object execution environment has behaviors for object persistence and local/remote transparency whose operation on an object depends on the object's location, e.g., machine and process. The MTS object execution environment has thread synchronization, automatic transactions, just-in-time object activation, declarative security and resource pooling behaviors that also are specific to an object's locality or other environment aspects that pertain to the object.

Brief Summary Text - BSTX (17):
The MTS system also implicitly associates a system-provided context object with each component application object hosted in the MTS execution environment. The context object encapsulates certain properties (e.g., a creator identity, a transaction, an **activity**, and security properties) that establish a "context" for the component application object within the MTS execution environment, and control certain of the environment's behaviors (e.g., automatic transactions, declarative security, etc.). However, the MTS object-context objects encapsulate properties specific to the domain-specific behaviors supported by the MTS system. Again, no structure is provided to readily extend the MTS object-context objects to support new domain-specific behaviors.

Brief Summary Text - BSTX (19):
The present invention introduces contexts, policies, policy makers and

activators that operate as general, extensible structure for automatic services to extend an object execution environment of an object-oriented system with domain-specific behaviors. According to an embodiment of the invention illustrated herein, independent aspects (e.g., threading model, declarative security, activity, transaction, etc.) of the object execution environment that are associated with objects are termed "domains." A group of one or more component application objects in the execution environment that share a common set of domains (i.e., are at an intersection of domains in the environment) are said to be in a "context." The illustrated embodiment represents the context as a context object that contains an ordered list of context property objects. The context property objects represent the domains and may be shared by more than one context.

Detailed Description Text - DETX (9):

A user may enter commands and information into the computer 20 through a keyboard 40 and pointing device, such as a mouse 42. Other input devices (not shown) may include a microphone, joystick, game pad, satellite dish, scanner, or the like. These and other input devices are often connected to the processing unit 21 through a serial port interface 46 that is coupled to the system bus, but may be connected by other interfaces, such as a parallel port, game port or a universal serial bus (USB). A monitor 47 or other type of display device is also connected to the system bus 23 via an interface, such as a video adapter 48. In addition to the monitor, computers typically include other peripheral output devices (not shown), such as speakers and printers.

Detailed Description Text - DETX (22):

Once the client of the COM object 60 has obtained this first interface pointer of the COM object, the client can obtain pointers of other desired interfaces of the component using the interface identifier associated with the desired interface. COM+ defines several standard interfaces generally supported by COM objects including the "IUnknown" interface. This interface includes a member function named "QueryInterface( )." The "QueryInterface( )" function can be called with an interface identifier as an argument, and returns a pointer to the interface associated with that interface identifier. The "IUnknown" interface of each COM object also includes member functions, "AddRef( )" and "Release( )", for maintaining a count of client programs holding a reference (e.g., an interface pointer) to the COM object. By convention, the "IUnknown" interface's member functions are included as part of each interface on a COM object. Thus, any interface pointer that the client obtains to an interface of the COM object 60 can be used to call the QueryInterface function.

Claims Text - CLTX (14):

14. In a runtime environment, where components execute in a context including a set of context-specific behaviors, a method comprising: receiving a reference request from a component in a first context; determining that the reference request is for a component of a second context; determining a first set of context-specific behaviors to enforce on cross-context requests made by the component in the first context to the component in the second context; returning to the component in the first context, a behavioral monitoring

reference to the component in the second context; receiving a cross-context service request from the component in the first context using the behavioral **monitoring** reference; and determining that the cross-context service request violates a context-specific behavior contained in the determined first set of context-specific behaviors.

Claims Text - CLTX (15):
15. The method of claim 14, wherein the first set of context-specific behaviors enforced on a cross-context service request made using the behavioral **monitoring** reference, is the union of the context-specific behaviors of the first and second contexts.

Claims Text - CLTX (16):
16. The method of claim 14, wherein a specific context-specific behavior is **monitored** by a component.

Claims Text - CLTX (20):
20. The method of claim 14, further comprising: receiving a call-back reference request for the component in the second context to make call-back requests to the component in the first context; determining a second set of context-specific behaviors to enforce on cross-context requests made by the component in the second context to the component in the first context; returning to the component in the second context, a behavioral **monitoring** call-back reference to the component in the first context; receiving a cross-context service request from the component in the second context using the behavioral **monitoring** call-back reference; and determining that the cross-context service request violates a context-specific behavior contained in the determined second set of context-specific behaviors.

Claims Text - CLTX (22):
22. The method of claim 21 wherein context-specific behaviors are **monitored** by context specific-behavioral components and a determination that a cross-context service request violates a context-specific behavior is determined through data sharing between a context-specific behavioral component in the first context and a context-specific behavioral component in the second context.